Laboratorio di Pensiero Computazionale

Simone Zennaro – simone.zennaro@unipd.it

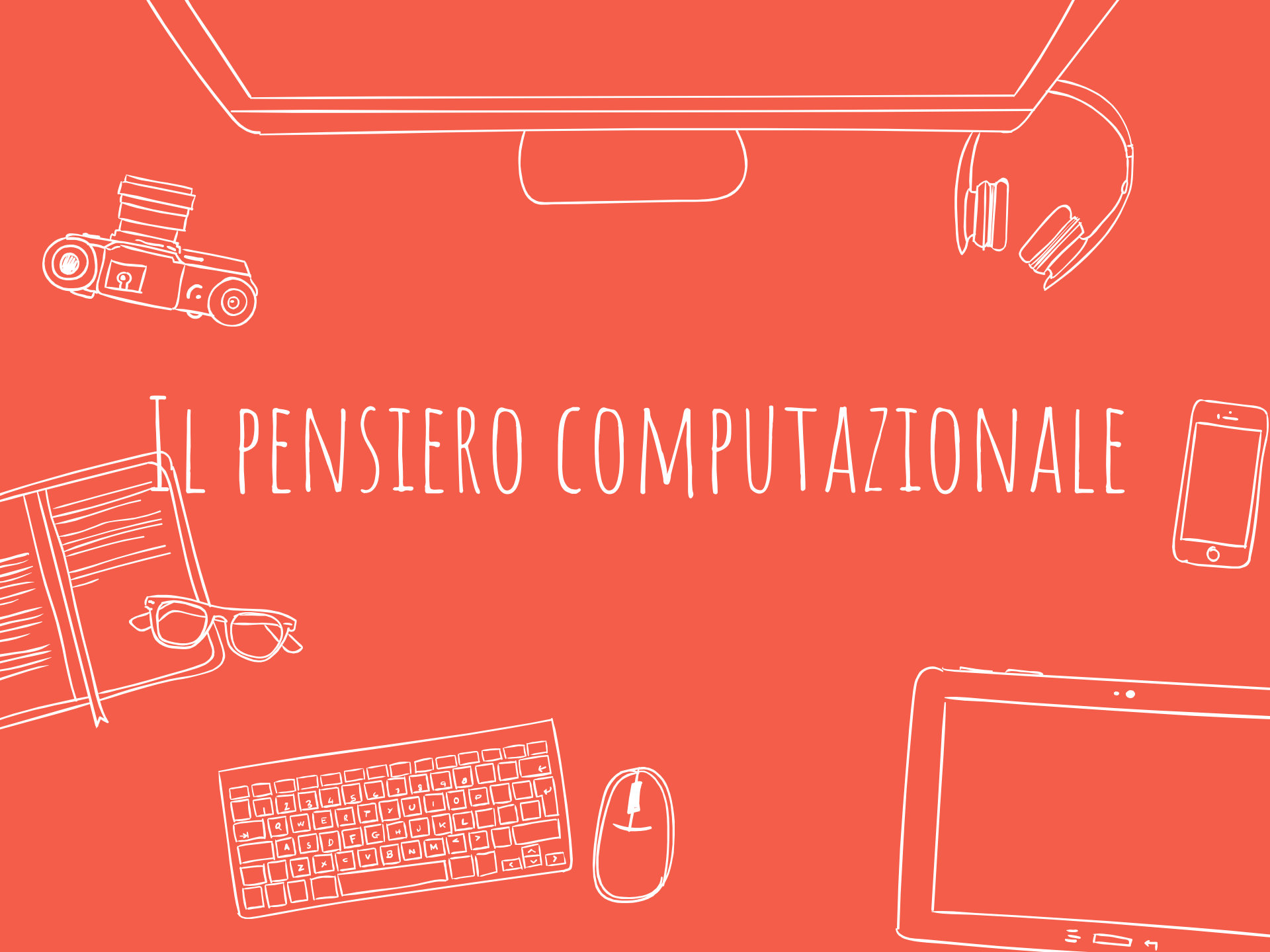
Nella lezione di oggi

- ✕ Introduzione ai sistemi di elaborazione
- ✕ Il pensiero computazionale
- ✕ Introduzione agli algoritmi
- ✕ Introduzione a Scratch
- ✕ Esercizi

Perché questo corso?

- Per imparare ad affrontare i problemi in modo analitico
- Per comprendere le basi e la necessità di programmare
- Per darvi degli strumenti utili ad affrontare i corsi che seguiranno
- Al termine di questo corso sarete in grado di scrivere ed eseguire il vostro primo programma in Python

Il pensiero computazionale



Il pensiero computazionale

- Il pensiero computazionale è un modo di affrontare e risolvere problemi ispirato ai principi dell'informatica, ma che si applica ben oltre la programmazione.

- In pratica è la capacità di scomporre un problema complesso in parti più semplici, individuare schemi ricorrenti, stabilire regole chiare e costruire una sequenza di passi logici (algoritmi) per arrivare a una soluzione efficace e generalizzabile.

Il pensiero computazionale

- Si può riassumere in alcune abilità fondamentali:
  - **Decomposizione** → dividere un problema complesso in sottoproblemi più gestibili.
  - **Riconoscimento di schemi** → individuare somiglianze o strutture ricorrenti tra problemi diversi.
  - **Astrazione** → concentrarsi sugli elementi essenziali, ignorando i dettagli superflui.
  - **Algoritmi** → progettare sequenze di passi chiari e ordinati per risolvere il problema.
  - **Valutazione** → riflettere sull'efficacia e l'efficienza della soluzione, migliorandola se necessario.

Il pensiero computazionale

- ✕ Non significa “pensare come un computer”, ma piuttosto **usare strumenti logici e sistematici per ragionare su problemi** in qualsiasi campo: dalla matematica alla biologia, dall’ingegneria alla vita quotidiana (per esempio, organizzare una ricetta di cucina come sequenza di passi ripetibili è già pensiero computazionale).

Dove?

- ✕ Cuocere una pasta *(sequenza di istruzioni)*
- ✕ Scegliere una cassa al supermercato *(minimizzazione)*
- ✕ Decidere se bere un altro spritz *(risorse disponibili)*

...E ANCORA, DOVE?

- ✕ Decidere se prendere l'autostrada *(compromesso tra costi e tempi)*
- ✕ Preparare lo zaino *(prefetching e caching)*
- ✕ Decidere quando smettere di noleggiare gli sci, e comprarne un paio *(algoritmi online)*

Procedura

1. Identificare il problema
2. Scomporre il problema in sottoproblemi
3. Esplorare le soluzioni per ogni sottoproblema e scegliere la migliore

È possibile individuare una **procedura** di individuazione di una soluzione del problema.

IMMAGINA DI DOVER PREPARARE LA VALIGIA PER UN VIAGGIO...

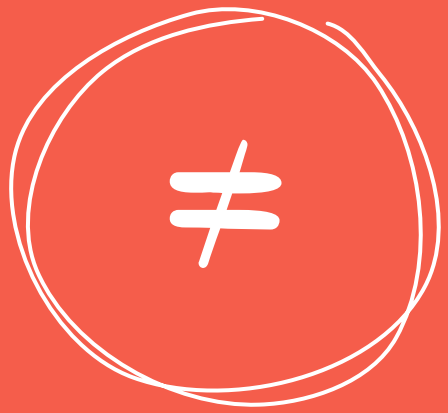
Se lo affronti con **pensiero computazionale**, ragioni così:

- **Decomposizione**
  - Dividi il problema in sottoproblemi: vestiti, oggetti per l'igiene, documenti, dispositivi elettronici.
- **Riconoscimento di schemi**
  - Ti ricordi che anche le altre volte che sei partito avevi bisogno di una lista simile (pantaloni, magliette, caricatore...).
- **Astrazione**
  - Ti concentri solo sugli oggetti davvero necessari per il tipo di viaggio (se vai al mare non serve la giacca pesante).
- **Algoritmo**
  - Definisci una sequenza ordinata di passi:
    - Prendere i vestiti.
    - Preparare i prodotti per l'igiene.
    - Controllare i documenti.
    - Aggiungere dispositivi e caricabatterie.
- **Valutazione**
  - Ti chiedi: "Ho tutto quello che serve? Posso ottimizzare lo spazio nella valigia?" e, se necessario, migliori il metodo.

Devi calcolare la media dei voti di uno studente

- **Decomposizione**
  - Il problema si divide in: raccogliere i voti, sommarli, dividerli per il numero di voti.
- **Riconoscimento di schemi**
  - Ti accorgi che la procedura è la stessa per qualsiasi insieme di numeri, non solo per i voti.
- **Astrazione**
  - Non ti interessa *quali* siano i voti (italiano, matematica, inglese...), ma solo i numeri.
- **Algoritmo**
  - Scrivi i passi:
    - Prendi l'elenco dei voti.
    - Somma tutti i voti.
    - Conta quanti voti ci sono.
    - Dividi la somma per il numero di voti.
- **Valutazione**
  - Controlli se il risultato ha senso: se i voti sono 6, 7 e 8, la media non può essere 15 o 2 → quindi verifichi l'accuratezza.

Programmazione



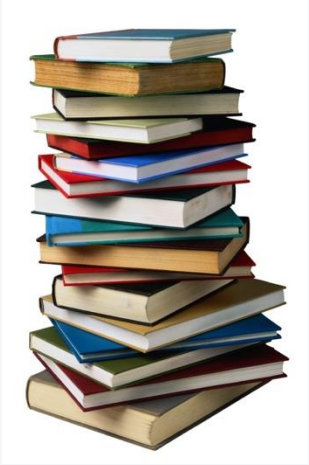
Pensiero computazionale

Relazione tra coding e pensiero computazionale

"Il pensiero computazionale è il processo mentale che sta alla base della formulazione dei problemi e delle loro soluzioni così che le soluzioni siano rappresentate in una forma che può essere implementata in maniera efficace da un elaboratore di informazioni sia esso umano o artificiale"

La programmazione è uno **strumento** importante per allenare il pensiero computazionale.

Informazione digitale: tutto diventa ‹‹bit››



Testo, Musica, Video, Immagini: tutto viene trasformato in una sequenza di 0 e 1

01100100

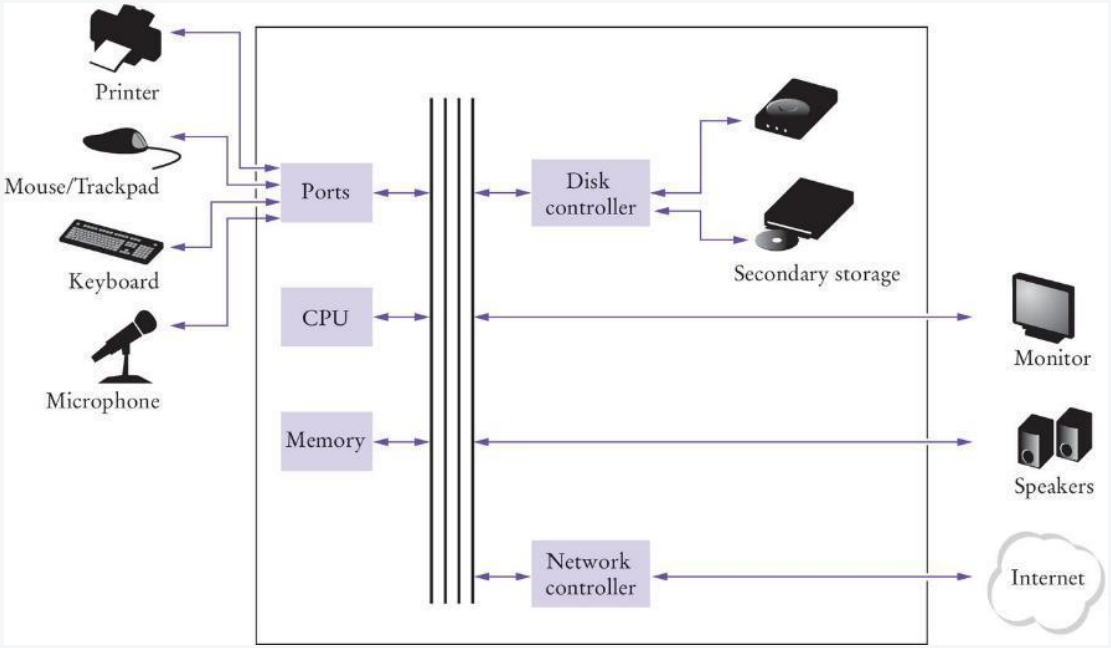
Hardware e Software

- Un calcolatore elettronico è composto di due parti:
  - **Hardware**: la component fisica, consiste di dispositivi elettronici e parti meccaniche, magnetiche ed ottiche
  - **Software**: la parte <<intangibile>>, consiste di:
    > Programmi: le istruzioni che dirigono l'hardware
    > Dati: le informazioni su cui lavorano programmi – spesso memorizzate nei file

Hardware

- Hardware comprende gli elementi fisici di un sistema di elaborazione o Esempi: monitor, mouse, memoria esterna, tastiera, ...
- La central processing unit (CPU) controlla l'esecuzione del programma e l'elaborazione di dati
- I dispositivi di memoria comprendono la memoria interna (RAM) e la memoria secondaria (Hard disk)
- I dispositivi di Ingresso / Uscita (Input / output) permettono all'utente di interagire con il computer (mouse, tastiera, stampante, schermo, ...)

Vista semplificata dell'hardware di un PC



Software

- Software viene tipicamente sviluppato sotto forma di «programma applicativo» (App)
  - Microsoft Word è un esempio di software
  - I Giochi elettronici sono software
  - I sistemi operativi ed i driver dei dispositivi sono anch'essi software
- Il Software è una sequenza di istruzioni e decisioni implementata in qualche linguaggio e tradotta in una forma che possa essere eseguita nel computer. I computer eseguono istruzioni molto semplici in rapida successione.

Programmi

Un programma per computer indica al computer la sequenza di passi necessaria a completare un determinato compito

- Il programma consiste di un (elevatissimo) numero di istruzioni primitive (semplicissime)
- I computer possono eseguire un ampio spettro di compiti perché possono eseguire diversi programmi
  - Ciascun programma è progettato per indirizzare il computer affinché lavori su un compito specifico
- Programmazione:
  - È l'atto (e anche l'arte) di progettare, implementare e verificare (testare) i programmi

Eseguire un programma

- Le istruzioni di un programma ed i relativi dati (come testi, numeri, audio o video) sono memorizzati in formato digitale
- Per eseguire un programma, questo deve essere portato in memoria, dove la CPU lo possa leggere
- La CPU esegue il programma un'istruzione per volta
  - Il programma può anche reagire agli input provenienti dall'utente
- Le istruzioni e gli input dell'utente determinano l'esecuzione del programma
  - La CPU legge i dati (compreso l'input utente), li modifica e li riscrive nuovamente in memoria, sullo schermo, o sulla memoria di massa

Assembly

```
MONITOR FOR 6802 1.4          9-14-80  TSC ASSEMBLER  PAGE    2


C000                    ORG     ROM+$0000 BEGIN MONITOR
C000 8E 00 70  START    LDS     #STACK

               ****************************************
               * FUNCTION: INITA - Initialize ACIA
               * INPUT: none
               * OUTPUT: none
               * CALLS: none
               * DESTROYS: acc A

0013           RESETA   EQU     %00010011
0011           CTLREG   EQU     %00010001

C003 86 13     INITA    LDA A  #RESETA   RESET ACIA
C005 B7 80 04           STA A  ACIA
C008 86 11              LDA A  #CTLREG   SET 8 BITS AND 2 STOP
C00A B7 80 04           STA A  ACIA

C00D 7E C0 F1           JMP    SIGNON    GO TO START OF MONITOR

               ****************************************
               * FUNCTION: INCH - Input character
               * INPUT: none
               * OUTPUT: char in acc A
               * DESTROYS: acc A
               * CALLS: none
               * DESCRIPTION: Gets 1 character from terminal

C010 B6 80 04  INCH     LDA A  ACIA      GET STATUS
C013 47                 ASR A            SHIFT RDRF FLAG INTO CARRY
C014 24 FA              BCC    INCH      RECIEVE NOT READY
C016 B6 80 05           LDA A  ACIA+1    GET CHAR
C019 84 7F              AND A  #$7F      MASK PARITY
C01B 7E C0 79           JMP    OUTCH     ECHO & RTS

               ****************************************
               * FUNCTION: INHEX - INPUT HEX DIGIT
               * INPUT: none
               * OUTPUT: Digit in acc A
               * CALLS: INCH
               * DESTROYS: acc A
               * Returns to monitor if not HEX input

C01E 8D F0     INHEX    BSR    INCH      GET A CHAR
C020 81 30              CMP A  #'0       ZERO
C022 2B 11              BMI    HEXERR    NOT HEX
C024 81 39              CMP A  #'9       NINE
C026 2F 0A              BLE    HEXRTS    GOOD HEX
C028 81 41              CMP A  #'A
C02A 2B 09              BMI    HEXERR    NOT HEX
C02C 81 46              CMP A  #'F
C02E 2E 05              BGT    HEXERR
C030 80 07              SUB A  #7        FIX A-F
C032 84 0F     HEXRTS   AND A  #$0F      CONVERT ASCII TO DIGIT
C034 39                 RTS

C035 7E C0 AF  HEXERR   JMP    CTRL      RETURN TO CONTROL LOOP
```

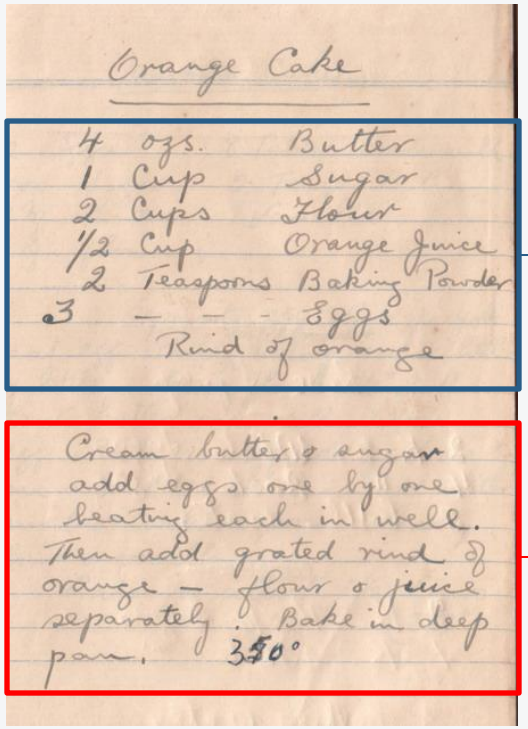
Il linguaggio assembly è un linguaggio di programmazione molto simile al linguaggio macchina, pur essendo differente rispetto a quest'ultimo

Assembly ≠ Assembler

Erroneamente viene spesso chiamato "assembler", ma quest'ultimo termine identifica solo il programma "assemblatore" che converte il linguaggio assembly in linguaggio macchina. Le istruzioni del linguaggio assembly corrispondono alle istruzioni che il processore può eseguire.

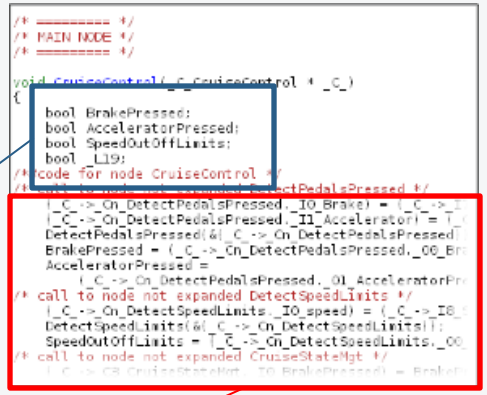
Assembly x86

Cucinare vs Programmare



Ingredienti/dati

Istruzioni



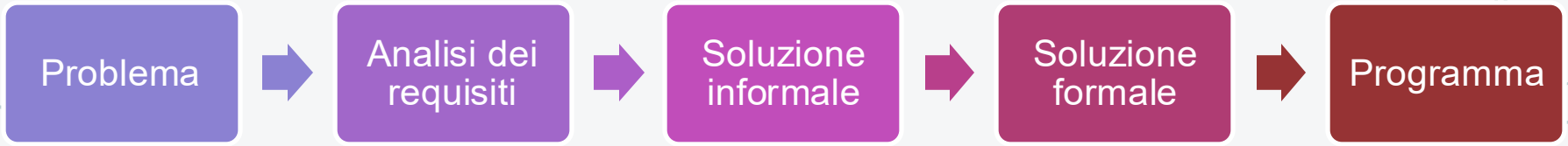
Cosa vuol dire "programmare"?

- La programmazione consiste nello scrivere un «documento» (file sorgente) che descrive la soluzione al problema considerato
- In generale, "la" soluzione a un dato problema non esiste
  - La programmazione consiste nel trovare la soluzione più efficiente ed efficace per il problema, secondo opportune metriche

Cosa vuol dire 'programmare'?

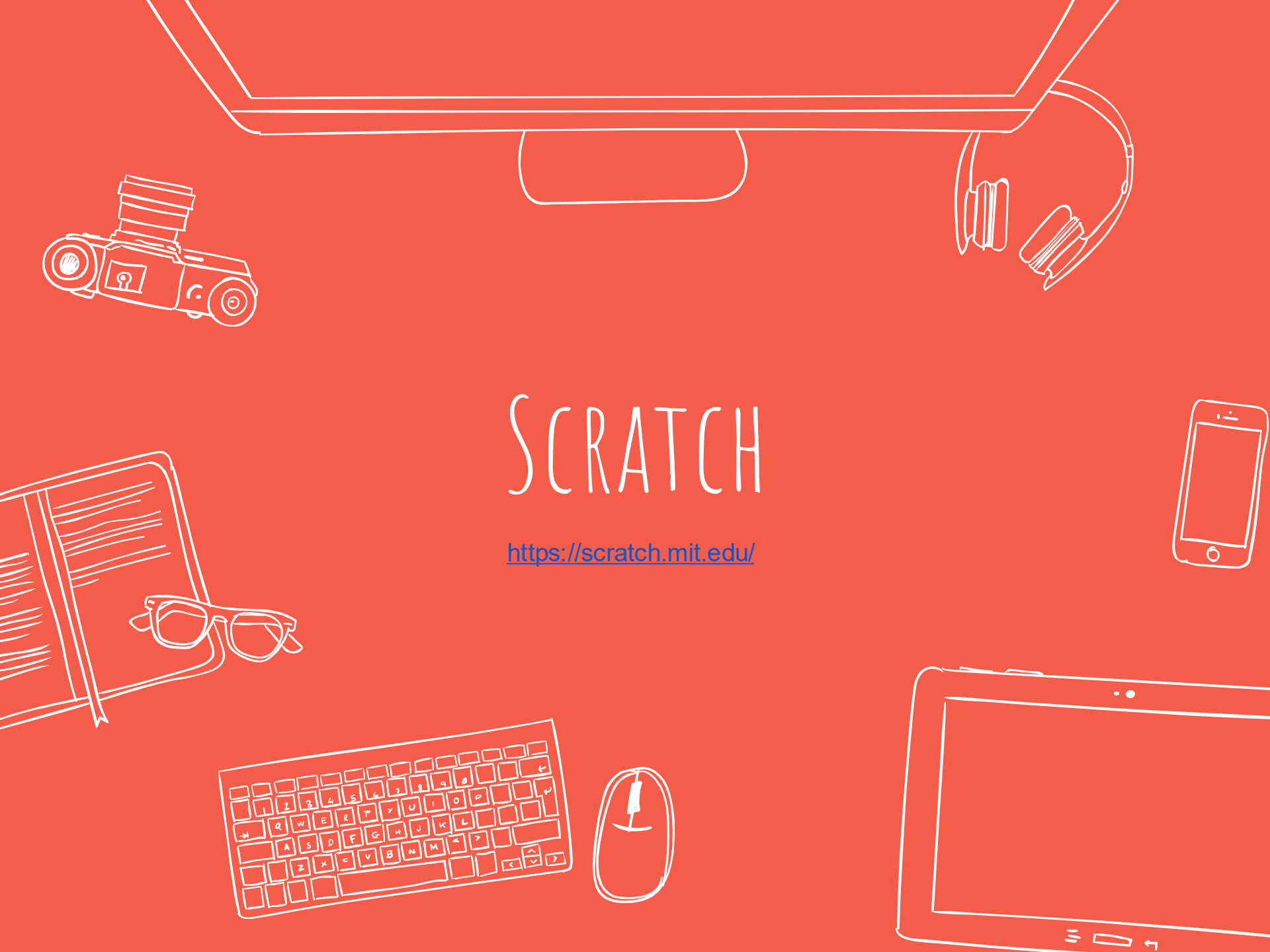
- ✕ Programmare è un'attività 'creativa'
  - Ogni problema è diverso da ogni altro problema
  - Non ci sono soluzioni sistematiche/analitiche o soluzioni universali
- ✕ Programmare è un'operazione complessa
  - Un approccio diretto è impraticabile o comunque molto difficile per compiti non banali

Sviluppo di un programma



CODE.ORG

https://studio.code.org/courses/express-2025/units/1

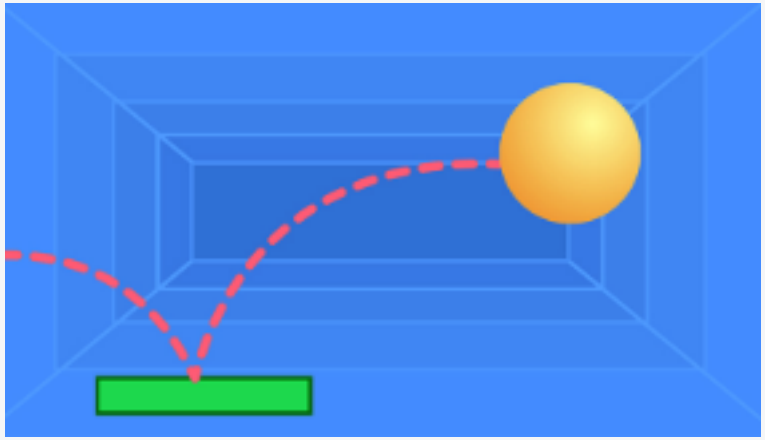


Scratch

https://scratch.mit.edu/

PONG

https://scratch.mit.edu/projects/editor/?tutorial=pong



Algoritmo

Un algoritmo descrive una sequenza di passi con le seguenti caratteristiche

- **Deterministico**
  - cioè quando partendo dagli stessi dati in input, si ottengono i medesimi risultati in output;
- **Non ambigua**
  - Non vi possono essere delle assunzioni sulla conoscenza necessario per eseguire l'algoritmo
  - L'algoritmo usa istruzioni precise
- **Eseguibile**
  - L'algoritmo deve poter essere svolto
- **Finito**
  - cioè quando è costituito da un numero finito di istruzioni e presenta una fine;

Problem Solving: Progettazione di algoritmi

- Gli algoritmi sono semplicemente dei piani operativi
  - Piani dettagliati che descrivono i passi per risolvere un problema specifico
- Alcuni li conosciamo già
  - Calcolare l'area di un cerchio
  - Trovare la lunghezza dell'ipotenusa di un triangolo
- Alcuni sono più complessi e richiederanno più passi
  - Risolvere un'equazione di 2° grado
  - Trovare il MCD tra due numeri
  - Calcolare la traiettoria di un razzo

Algoritmi nella vita quotidiana

1. Metti l'acqua
2. Accendi il fuoco
3. Aspetta
4. Se l'acqua non bolle torna a 3
5. Butta la pasta
6. Aspetta un po'
7. Assaggia
8. Se è cruda torna a 6
9. Scola la pasta

Chi si è accorto che manca il sale?

Trova il minimo

Trova il minimo

Dobbiamo descrivere dettagliatamente il *processo* attraverso il quale una persona trova il più piccolo elemento di una lista. La domanda alla quale rispondere è: *Cosa sta davvero facendo una persona quando deve trovare il più piccolo elemento di una lista?*

Configurazione iniziale e Regole

- Useremo delle carte a faccia in giù su un tavolo per rappresentare una lista di elementi. Si comincia con 6 carte a faccia in giù, disposte in riga.
- Ogni carta, quando posta sul tavolo, deve essere a faccia in giù.
- Quando si agisce come una macchina, si può raccogliere una carta con una delle due mani. Ogni mano può raccogliere e tenere una carta alla volta. Non è possibile tenere più carte con la stessa mano.
- Si possono guardare i valori delle carte che si hanno in mano. I valori delle carte presenti nelle due mani possono essere confrontati tra loro (per esempio per capire quale dei due è il più grande).
- Si può mettere giù una carta sul tavolo (a faccia in giù). Tuttavia, una volta che la carta è a faccia in giù sul tavolo, si deve dimenticare tutto su di essa (ovvero non ci si può ricordare il suo valore o la posizione che occupa nella lista). In altre parole, non si possono memorizzare informazioni riguardanti carte che sono a faccia in giù. Notare che la mente umana fa questo in automatico, ma nella descrizione del procedimento da seguire non è permesso usare tali informazioni.

Compito

- Obiettivo: l'algoritmo deve terminare in modo chiaro. L'ultima istruzione dovrebbe essere "Carta trovata!" e si dovrebbe avere in mano la carta con valore minimo.
- L'algoritmo dovrebbe essere scritto in modo indipendente dal numero di carte presenti nella riga. In altre parole, dovrebbe teoricamente funzionare con qualsiasi numero di carte, siano esse 1, 6 o 1000.
- Scrivere l'algoritmo come una chiara lista di istruzioni. Le istruzioni che vengono usate possono fare riferimento al valore delle carte, alle mani della persona ecc., ma si deve inventare un modo sistematico per trovare la carta con valore minore. Suggerimento: le istruzioni dovrebbero essere numerate, in modo da rendere la sequenza chiara e non ambigua.